A COLLABORATIVE ENVIRONMENT TO LEARN PROGRAMMING

G. Bizzarri, L. Forlizzi and F. Ricci
Dipartimento di Informatica, University of L'Aquila, via Vetoio, 67010, L'Aquila, Italy

Keywords: e-Learning, Collaborative Programming, Wiki, Teaching Tool.

Abstract: Students taking their first steps in the programming world need to find resolved examples, compare their solutions to well-know problems and to understand the errors that are returned by a compiler. We have planned to create a wiki for source code and give to the students an e-learning platform that allow them to write code in a collaborative way, integrated with a technology to compile the source code written in different programming languages, to interpret errors returned by the compiler and to show them by a virtual tutor speaking in their national language and that use the natural language of everyday life. It helps to understand the errors, where they were committed and how fix them.

1 INTRODUCTION

In the Internet era, young people often have a considerable experience with Information and Communication Technologies (ICTs). However, they are especially familiar with easy-to-use technologies like smartphones, entertainment devices, web browsers. Their skills are often restricted to the ability to interact, quickly and effectively, with such devices by means of graphical user interfaces.

The use of more sophisticated technological tools, with interfaces not designed for ease of use, nor based on the usual graphical metaphors proves to be far more difficult for the so-called "digital natives" (Prensky, 2001). The confidence in the use of many digital devices is accompanied by a profound misunderstanding of the world of technology within young people: they use resources, systems and devices of which they ignore working principles and inner details, thus regarding them the same way as magical objects (Longo, 2009).

Despite this profound limitation, the attitude and the tendency to use computers and communication devices is an important skill that one should try to exploit to reach deeper levels of knowledge and comprehension. A promising strategy to achieve this aim is to use take advantage of one relevant consequence of the use of communication devices, namely the dense network of social relations that these devices allow to establish.

Blended e-learning is increasingly emerging as the most effective use of information technology in education. In blended courses, a variety of technological tools such as discussion boards, chats, wikis, and blogs are employed to facilitate discussion and interaction (Lord. and Lomicka, 2008). Regular use of these tools is important to the development of a learning community and to the promotion of learning and interaction at a distance; such tools, rather than leading to cold and dehumanizing contact, can, in fact, promote a sense of community (Rovai and Jordan, 2004).

In this work we present an e-learning tool devoted to the teaching of programming at an introductory level that promote the construction of a learning community and tries to achieve educational benefits from the interaction between learners.

Novice programmers (and sometimes experienced ones) tend to search for a lot of carried out examples. We regard this as an inherently positive attitude, since we believe learning-by-example to be an appropriate strategy to grasp the basics of programming. Indeed, we think that at an introductory level, the main difficulty is not to understand the concepts upon which programming languages are based (i.e., in the case of imperative languages, variable, expression, instruction) but rather in acquiring the ability to use and combine simple concepts and language constructs to put together working programs.

Bizzarri G., Forlizzi L. and Ricci F..
A COLLABORATIVE ENVIRONMENT TO LEARN PROGRAMMING.
DOI: 10.5220/0003964904710476
In *Proceedings of the 4th International Conference on Computer Supported Education* (ESEeL-2012), pages 471-476
ISBN: 978-989-8565-07-5
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

The problem is that novice programmers who are digital natives tend to search for program examples on the internet, often finding programs that have low quality or are too much sophisticated for their current skill level. The tool we propose allow the construction of program examples repositories to satisfy the needs of novice programmers. In order to control the quality of programs, we employ two main mechanisms:

- use of compilers to check correctness and other properties of programs submitted to the repository;
- collaborative writing, development and maintenance of the programs.

A key element for the success of this approach, is to motivate student to participate in the collaborative effort to build and extend program collections stored in the system. A major obstacle that discourages active participation of students is the lack of familiarity in the use of compilers, which, facing a series of failures in the compilation of programs, can lead to mistrust and fear to submit one's work to the judgement of the machine. To address this problem we provide the system with a *virtual tutor*, i.e., a software agent which provides guidance in natural language and assists the students to correct programming errors. Usually compilers, following the detection of errors, emit rather brief messages using highly technical terminology, which is hardly understandable by novice programmers. Also, compilers rarely provide some indication on how to correct errors.

Moreover, almost all compilers return an output in English language only. This fact creates additional difficulties to students less familiar with this language.

The program repository thus becomes the center of a rich learning environment, which also allows online creation of small programs in a code highlighter editor. To allow collaborative writing of programs, our system is based on a wiki system. In preparation for future integration with other learning resources, we chose to embed our system in a Learning Contents Management System (LCMS).

The rest of the paper is organized as follows. Section 2 describes in more detail the learning environment we propose. In section 3 we present some teaching activities in which we are currently experimenting the usefulness of the system. Finally Section 4 gives some hint on our plans for future developments and Section 5 draws some concluding remark.

2 THE ENVIROMENT

The system is based on two main components: a central core that handles all kinds of content and allows the fruition to the students and the virtual tutor which gives an added value to the student who uses the system. We decided to build the repository inside an LCMS in order to take advantage of its infrastructure for the presentation of contents. Among the many LCMS available, we chose Moodle on the ground of its widespread use: the number of its active sites grew from about 1.000 on early 2004 to 73.000 at the end of 2011, continuing its run with an average of 2.000 new activations per month.

2.1 Wiki

The program repository is built around a wiki system, that allows to share, exchange and store the source code in a totally collaborative way. For any user of the system (registered or anonymous) it is possible to:

- add content;
- edit content;
- delete content;
- check the version history for a single content;
- check the differences between two or more versions of a single content;
- restore a previous version of a single content.

These are basic features of any wiki, provided in our system by extending the standard wiki module present in Moodle, to better handle any kind of source code.

In order to promote a rapid growth of the code base in the repository, in the current version of the system all users are supposed to have the same level of expertise.

Therefore each user's additions, changes or deletions to existing programs take place with no need of approval by other users. In light of this strategy, a very important feature of the repository is the preservation of the succession of changes to the stored programs.

This allows to restore old versions of the programs in case of mistakes (or vandalism). Moreover it allows to compare multiple versions of a source code in order to analyze how it evolved through a sequence of modifications in terms of error correction and program quality. To the latter aim, the history of a program also contains diagnostic messages emitted by the compiler for each single translation performed.

2.2 User Stats

In an e-learning system, in addition to learning tools and objects, it is important to have reporting tools allowing students to track their progress having an evaluation, teachers to monitor the development of individuals and of the group.

For this reason the system is equipped with a tracker for the operations in order to store statistics about translations performed and errors found. Errors are grouped into macro categories according to their type and their severity. For each translation a record is inserted into the database containing the id of the page, the user id and the number of errors differentiated by type. Such an organization allows, by intersecting the data, to generate a variety of statistics like:

- % of compilation with errors for student;
- % of errors for student grouped by typology;
- % of compilation with errors for program;
- % of compilation with errors for program grouped by typology;
- % of compilation with errors for course;
- % of compilation with errors for course grouped by typology;
- list of students who make more errors;
- list of students who make more errors grouped by typology;
- list of pages where are made more errors;
- list of pages where are made more errors grouped by typology;

These data can be used for statistical purposes: for example to perform comparisons between students and work groups, or any act aimed to reinforce topics subject of major errors.

2.3 History

Gaston Bachelard (1977) believes that the error is positive, normal and useful, and endorses his theory by claiming that the error is not an obstacle to knowledge, rather this is characterized as a perspective of errors corrected.

In this perspective, we think that students can benefit a lot from the use of a tool that allow them to inspect the history of a given source code, to study how errors were committed and were propagated, and to highlight the differences between successive versions.

Both the teacher and the students can access these data to identify the issues and situations in which mistakes are made and to analyze how students' knowledge and ability to program improve over time.

We consider it particularly important that students have available tools to analyze their own progress and error, because this helps to improve critical thinking, the ability to self-evaluation and the engagement in the use of the system beyond the scheduled lectures.

Another important reason to provide historical data is code reuse.

When programming, one often finds himself in front of errors or problems that have already been addressed before, but for which one does not know the solution, either because one forgot or because it was provided by another person.

The repository can be used as a source of ready to use solutions, explained in detail and validated both by means of automatic correction provided by the system, either by the sieve of other users.

2.4 Module Compile and Report

In this section we describe in more details the tools that allow to compile source code, which constitute the real core of the system.

After the creation or the update of a page containing source code, unlike a conventional wiki which just saves the content into a database, the compiling module comes into play. According to the specific programming language employed, it extracts the source code from the page and dispatch it to the appropriate compiler. If the compilation is successfully completed, no further operations will be performed. Otherwise, diagnostic messages emitted by the compiler are submitted to the virtual tutor.

The virtual tutor's role is to interpret the output of the compiler and to provide to the user a natural language description of errors, other problems, possible solutions, related topics

This poses two main difficulties:

- the need to know all the possible errors detectable by the compiler;
- the need to be able to recognize individual errors, despite the fact that different compilers usually emit completely different diagnostic messages for the same problems.

As regards to the first issue, the only solution is a total enumeration of all possibilities. This solution, although expensive, is the only that guarantee a complete feedback to users, a fact that we consider important for novice programmer.

To overcome the second difficulty, we employ a set of regular expressions and try to match them with diagnostics emitted by the compiler. The line number and the list of program objects involved are determined by the regular expression matching the diagnostic messages, and they provide very useful information to identify the roots of the problem.

2.4.1 Example

To clarify the concept we show the following example in C language.

```
void main()
{
int a[];
a[0]=5;
}
```

The output returned by the gcc compiler is the following:

```
7: error: array size missing in 'a'
```

This is the typical error of an array declaration where the size of the array is not specified. It would be sufficient to find a match with “error: array size missing” to recognize the error, but in addition at the beginning and at the end of the line there are very important information that can be shown to the user: the row number and the name of the array that has caused the mistake.

The regular expression we use to detect this class of errors is:

```
#([0-9]*): error: array size missing in
'([A-Za-z0-9_]*)'#i
```

Such a regular expression allows to perform the match and select the row number and the variable causing the error. The virtual tutor can now produce an appropriate message, more understandable by a novice programmer, like for instance “At line 7, you have declared the array ‘a’ without size. Each array must have a positive integer size specified between the brackets”. The message can be complemented by learning contents related to variable declarations and characteristics of the array data type.

3 CASE STUDIES

To assess the impact of the tool and methodology that we propose, based on collaborative programming, we are using them in a couple of teaching experiences.

3.1 Olympiads in Informatics Training

A special training course for the Olympiad in Informatics, has been used as a case study to test the effectiveness of this approach (Verhoeff, 2006).

For this experiment the repository was built using a previous version of our system which uses, instead of wiki pages, an expansion module for Moodle that we realized for this specific task. (Barbieri et al., 2011).

The expansion module provides additional capabilities to the LCMS:

- a revision control system for source codes that allows collaborative program development;
- a module responsible of compiling and reporting diagnostic messages to students;
- a test environment where successfully compiled programs are executed with sets of input instances created by the teacher;
- instruments to manage and organize work groups suitable for our needs.

Has been decided in fact to experiment the usage of this module in a special training course managed by teachers of the University of L’Aquila in collaboration with teachers of high school for the AAOII (Italian acronym for: Abruzzesi Trainings of Olympiad in Informatics.

The purpose of the training course is to prepare adequately, to the regional and national phases of the competition, students coming from different high schools who were selected having reported an high score in the first round of the competition, in which they faced a mixture of logical and mathematical puzzles and very simple programming tests.

The regional phase of the Olympiads in Informatics consists of a contest in which participants must create programs written in a programming language chosen from Pascal, C, C++ that solve algorithmic problems similar to those of National phase, but with lower difficulty.

By using the repository and the compile-and-report module, it was possible to extend the teaching activity outside the planned hours of lectures, allowing the students to be able to work from home and to verify the correctness of their own programs online.

The students had access to the repository, containing all the source code shown in class, which has evolved during the whole course, and were able to practice from home testing the proposed solutions to individual problems directly through the module included in the e-learning platform.

The test environment allowed students to check run-time correctness and efficiency of their programs by running them on selected input instances.

Tests instances were created by teachers in order to cover a wide range of possible input types, including very large instances and borderline cases.

Students were positively stimulated by the availability of a quick and precise feedback to their work.

The participants in the training course were asked some written comments on their learning experience to test the success of the initiative, the overall efficiency of the course and especially the usefulness of e-learning platform.

- 92% of students said they had visited the site at least once
- 75% said they had used the tools available to supplement the hours of frontal course;
- 83% of respondents would find it useful to use a platform like the one made available in this course to complement the regular school work;
- 75% appreciated the simulation of a competition conducted by the online delivery of elaborated and almost all of them would repeat the experience next year.

These data, although collected on a limited sample of students, have encouraged us to continue experimenting with this approach and gave us valuable information for subsequent developments.

3.2 A University-level Computer Science Course

The tool presented in this paper will be used in a first-year programming course starting in March 2012 in a University.

We plan to test benefits provided by the individual features of the tool through an evaluation of differences in students' performance against those of previous years in which the system was not available.

The evaluation and subsequent adjustments will be based on assessments of the course, compared to the corresponding in previous years, the objective results of the examinations, on a questionnaire submitted by students at the end of the course and on the analysis of the history of code assignments.

4 FUTURE DEVELOPMENTS

The system was developed to help students, especially the beginners in the programming world, but nothing prevents to extend the system with other tools that can help even the advanced programmers in order to make it more complete and increase the catchment area of the possible users.

4.1 Static Analysis Tools

One of the developments planned for future versions of our system is the integration of tools that perform static analysis of code.

We are currently in the process of reviewing several tools capable to check that a piece of code does not contain semantic anomalies and satisfies the particular characteristics of correctness.

A list, although incomplete, of the kind of checks we would like to offer, is the following:

- declaration of variables;
- initialization of arrays and strings;
- assignments to variables of different length or size;
- division by 0;
- precedence between operators;
- data types in comparisons between variables;
- correctness of boolean expression;
- initialization of memory variables;
- access to uninitialized memory variables;
- indices of array not positive or out of range;
- wrong identifier names;
- incorrect access to data structured;
- presence of infinite loops;
- presence of unreachable code.

In all these areas there are errors that usually escape standard analysis performed by compilers, and are found only at run-time. The identification of the cause of a runtime error is extremely difficult because very often there is not an exhaustive description.

It is therefore important to make a sophisticated analysis of the behavior of every single statement and declaration throughout the source code to try to discover the possible programming errors (even hypothetical), that would be unrecognized by the compiler and which could after cause problems at run-time.

4.2 Artificial Intelligence

So far we have mainly concentrated in errors recognition and in the identification of possible problems in the code in order to assist students in self-correction.

Another future development is to integrate the system with elements of artificial intelligence as an added value for teaching.

These elements will be integrated in the core as independent subsystems which, basing their work on several pieces of data stored by the history of the items and compilation statistics, will try to identify, for each user of the system, specific areas of his programming activity which seems to pose more challenges to him, allowing the tutor to suggest better solutions and more appropriate teaching material. Since our aim is to show suggestions based on the observations, the more promising approach is the knowledge-based reasoning in which an agent uses the data stored in the database as a knowledge base to perform meta-reasonings about them. (Bylander and Chandrasekaran, 1987)

To give a concrete example, imagine that, by examining the history of the repository, the system discovers that 6 students out of 10 have made at least once an error concerning the management of pointers, and 3 of them have made this type of errors with a very high frequency; it should be concluded that it may be appropriate for the teacher to review the topic with the class.

As a second example, imagine that there is a high percentage of errors in implementation or use of the Bubble Sort algorithm. The system will infer that it is appropriate to spend some time to revisit topic, and warn the teacher.

5 CONCLUSIONS

The main purpose of our approach is to improve students' critical thinking in solving programming problems. This can be achieved by pursuing the following more specific objectives:

- to help novice programmers to understand diagnostic messages emitted by compilers and the roots of their errors;
- to teach not only to recognize the mistakes but to learn from them;
- to teach the use of a revision control system to manage source code collections;
- to make students work in group and in a peer exchange (peer education) and to acquire the role of the tutor towards novices;
- to promote the use of technology as a means of to understand and not only to perform tasks.

REFERENCES

Bachelard, G. (1977). *La formulation de l'esprit scientifique.* Paris: Vrin.

Barbieri, A. and Bizzarri, G. and Forlizzi, L. (2011), Gruppi dinamici e compilazione on-line. In: Baldoni, M. and Baroglio, C. and Coriasco, S. and Marchisio, M. and Rabellino, S. *E-learning con Moodle in Italia: una sfida tra passato, presente e futuro* (pp 183–194). Torino: Seneca Edizioni.

Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2), 231–243.

Gokhale, A. A. (1995). Collaborative Learning Enhances Critical Thinking. *Journal of Technology Education,* 7(1).

Longo, G. O. (2009). Nascere digitali. Verso un mutamento antropologico?, *Mondo digitale*, 32, 3-20.

Lord, G. and Lomicka, L. (2008). Blended learning in teacher education: An investigation of classroom community across media. *Contemporary Issues in Technology and Teacher Education*, 8(2).

Prensky, M. (2001) Digital Natives, Digital Immigrants *On the Horizon,* 9(5), 1-6

Rößling, G. and Crescenzi, P. and Ihantola, P. and McNally, M. and Radenski, A. and Sànchez-Torrubia, M. G. (2010). Adapting Moodle to Better Support CS Education. *Proceedings of the 2010 ITiCSE working group reports*, 15-27.

Rovai, A. P. and Jordan, H. M. (2004). Blended learning and sense of community: A comparative analysis with traditional and fully online graduate courses. *The International Review of Research in Open and Distance Learning,* 5(2).

Verhoeff, T., (2006) The IOI is (not) a science olympiad. *Informatics in Education 5(1)*, 147-159.